

DOCUMENT RESUME

ED 302 234

IR 013 656

AUTHOR Heck, William P.; And Others
TITLE Guidelines for Evaluating Computerized Instructional Materials. Second Edition.
INSTITUTION National Council of Teachers of Mathematics, Inc., Reston, Va.
REPORT NO ISBN-0-87353-219-8
PUB DATE 84
NOTE 38p.
PUB TYPE Guides - Non-Classroom Use (055) -- Reports - Evaluative/Feasibility (142)

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS *Computer Assisted Instruction; *Courseware; Elementary Secondary Education; *Evaluation Criteria; Guidelines; Higher Education; *Instructional Effectiveness; *Instructional Material Evaluation
IDENTIFIERS Software Evaluation; *Software Selection

ABSTRACT

This document presents guidelines for evaluating computerized instructional materials. Included are reference sources, suggestions for evaluation criteria, guidelines for both software and documentation evaluation, and forms that can be used in the evaluation process. Five separately labeled reference lists contain citations for approximately 100 items. (EW)

* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *

ED302234

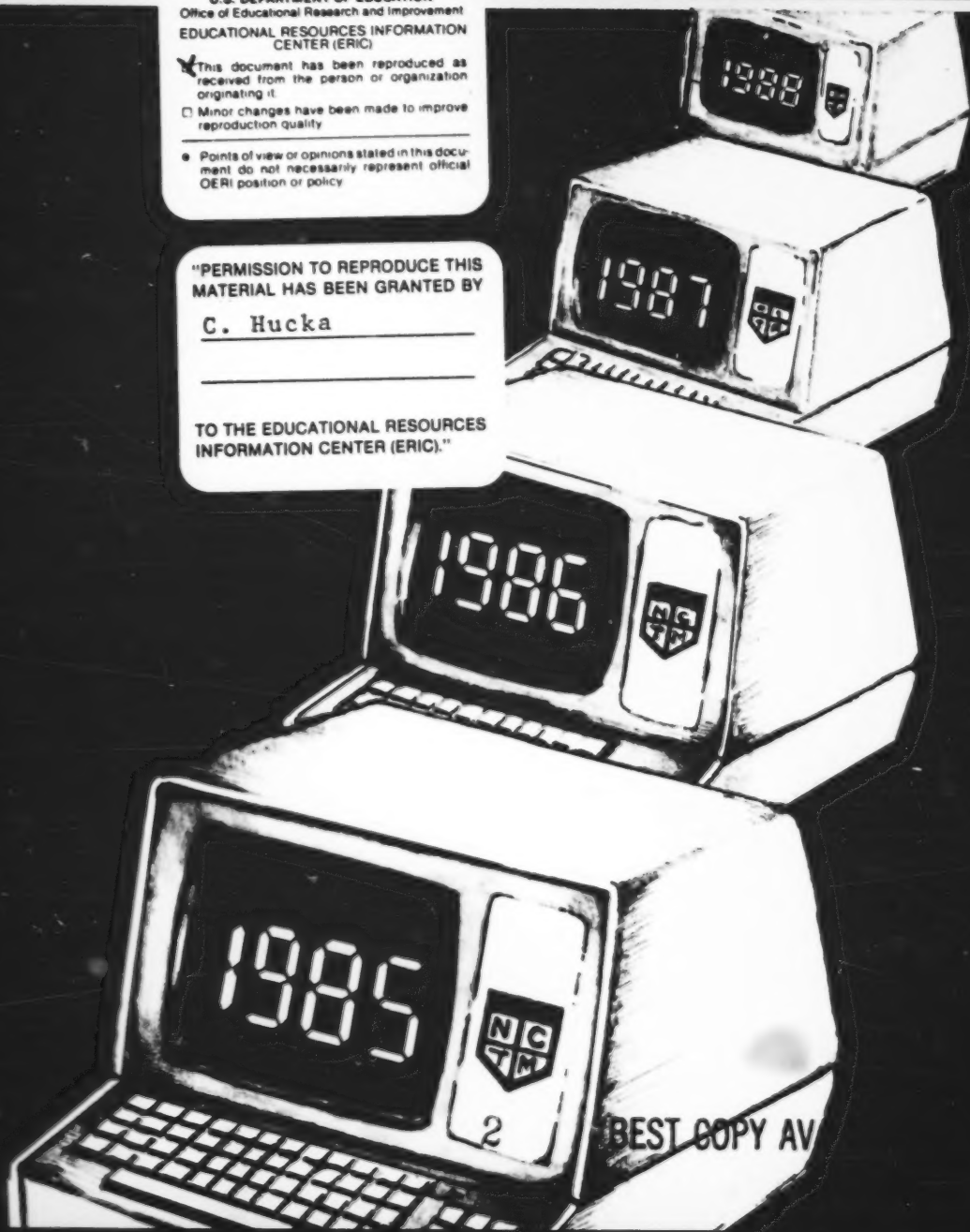
U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION
CENTER (ERIC)

- ☒ This document has been reproduced as received from the person or organization originating it.
- ☐ Minor changes have been made to improve reproduction quality.
- Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

"PERMISSION TO REPRODUCE THIS
MATERIAL HAS BEEN GRANTED BY

C. Hucka

TO THE EDUCATIONAL RESOURCES
INFORMATION CENTER (ERIC)."



IR013662

AN AGENDA FOR ACTION

Recommendations for School Mathematics of the 1980s

- 1 Problem solving must be the focus of school mathematics in the 1980s
- 2 The concept of basic skills in mathematics must encompass more than computational facility
- 3 **Mathematics programs must take full advantage of the power of calculators and computers at all grade levels**
- 4 Stringent standards of both effectiveness and efficiency must be applied to the teaching of mathematics
- 5 The success of mathematics programs and student learning must be evaluated by a wider range of measures than conventional testing
- 6 More mathematics study must be required for all students and a flexible curriculum with a greater range of options should be designed to accommodate the diverse needs of the student population
- 7 Mathematics teachers must demand of themselves and their colleagues a high level of professionalism
- 8 Public support for mathematical instruction must be raised to a level commensurate with the importance of mathematical understanding to individuals and society



National Council of Teachers of
Mathematics
1906 Association Drive, Reston, Virginia 22091

Guidelines for Evaluating Computerized Instructional Materials

prepared by

William P. Heck
Jerry Johnson
Robert J. Kansky
J. Richard Dennis

for the

**Instructional Affairs Committee
of the
National Council of Teachers of Mathematics**

TABLE OF CONTENTS

Section 0: A Guide to These Guidelines	1
Section 1: What, Why, and for Whom?	3
Section 2: Obtaining Software and Documentation	5
Section 3: Getting Hardnosed about Software: Guidelines for Software Review	10
Section 4: Communicating with Posterity: Guidelines for Software Documentation	18
Section 5: Asides to the Instructional Programmer	21
Section 6: Next Steps	23
Section 7: Extra Forms	26

National Council of Teachers of Mathematics

Copyright © 1981, 1984 by

The National Council of Teachers of Mathematics, Inc.

1906 Association Drive, Reston, Virginia 22091

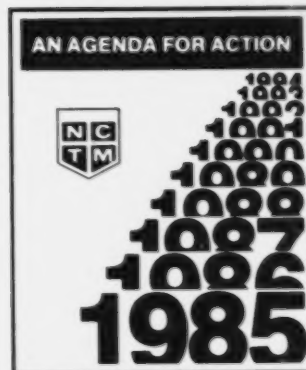
All rights reserved

Printed in the United States of America

Second edition

ISBN: 0-87353-219-8

Second printing 1986



PREFACE

These guidelines are offered to *all* educators interested in the instructional applications of computers. Their development has been marked by a continuous, conscious effort to address the needs of all educators irrespective of their areas of interest, subject matter, or present level of involvement with instructional computing.

The preparation of *Guidelines for Evaluating Computerized Instructional Materials* was carried out under the direction of the NCTM's Instructional Affairs Committee (IAC) as part of its effort to implement recommendation 3 of *An Agenda for Action* (see the inside front cover of this booklet). The writing and field testing were done by a special IAC subcommittee composed of Robert J. Kansky, *chairman* (Texas A & M University, College Station, Texas), William P. Heck (science and mathematics supervisor, Robbinsdale Area Schools, Minneapolis, Minnesota), and Jerry Johnson (Western Washington University, Bellingham, Washington). J. Richard Dennis (University of Illinois, Champaign-Urbana, Illinois) was a coauthor of the revised edition. The cover design was suggested by Earl Ockenga (Price Laboratory School, Cedar Falls, Iowa).

SECTION 0

A Guide to These Guidelines

What Follows 1?

The preparation of this resource presented three tasks: (1) to identify information about the location, selection, and evaluation of computerized materials that might be of interest to all educators, (2) to report that information in intellectually bite-sized pieces called *sections*, and (3) to arrange those sections in the one true order that would assure maximum understanding by each reader. With more than a little help from friends and colleagues, the authors have tried to meet the challenge of the first two tasks; the last task, however, proved too much. (Ah well, two out of three isn't bad. Given today's grade inflation, the performance probably rates a B+.) The reader should not infer that the sequence of section numbers corresponds to an optimum reading order. Section 2 may very well not be the logical successor to section 1. With one exception (section 5), the sections of this document should be of use to any educator who is concerned with the instructional application of computers. However, the best path to follow in confronting the issues of those sections depends, at least in part, on the reader's present status with regard to the instructional use of computers. The purpose of this section is to help you find the route appropriate to your interests and experiences and to suggest some useful side trips along the way.

Sign Up Here

Five "tour groups" that are defined below in terms of the instructional computing skills and goals of their members have been formed for readers. Each group has an itinerary. You are invited to join the one whose membership seems most compatible.

***Tour Group A.** Members of our group have had no experiences in programming a computer. Indeed, many of us have neither the time nor the inclination to pursue such experiences. Our interests lie in becoming intelligent users of computerized instructional materials.*

Itinerary:

1. Read section 1.
2. Visit a school or computer store for an hour or so of actual interaction with a computer. Ask a lot of questions to clarify what's in section 1.
3. Read sections 2 and 3.
4. Return to the school or computer store for more hands-on experience. Try to find examples of the types of programs and documentation described in sections 2 and 3.
5. Skim section 4; ignore section 5; peruse section 6.
6. Obtain and evaluate some materials, revisiting sections 2 through 4 as needed.

***Tour Group B.** All our members are experienced computer programmers. Many of us teach computer programming. We're interested in learning what characterizes instructional software. We may need to aid teachers who are trying to develop or purchase such software.*

Itinerary:

1. Skim sections 1 and 2.
2. Carefully read section 3.

3. Try classifying several available programs according to the uses described in section 3; evaluate at least one of those programs using the sample checklist.
4. Reread section 3.
5. Read sections 4 through 6.
6. Use sections 4 and 5 to guide students (and yourself) in preparing in-house documentation.
7. Rewrite a program so that it would receive a favorable review under the criteria you adopt from section 3.

Tour Group C. *Although some of our members have taught or could teach computer programming, our primary interest is in developing our own programs for instructional use in our own classrooms in subjects other than computer programming.*

Itinerary:

1. Skim section 1.
2. Read section 3. Use it to classify and evaluate some existing programs.
3. Read sections 4 and 5.
4. Develop an appropriate checklist (section 3) and documentation standards (sections 4 through 6).
5. Read sections 2 and 6.

Tour Group D. *Our members teach a variety of subjects with the help of computerized instructional materials that they have purchased or have been given. Most of us have no computer programming experience; those who have such experience do not have a strong interest in applying that experience to the initial development of computerized courseware. Our interest is in using the work of others to improve instruction in our subject-matter areas. This interest includes a wish to encourage our colleagues to make instructional uses of computers.*

Itinerary:

1. Read sections 1 through 4. (The programming members of the group may wish to skim section 5.)
2. Use section 3 to develop an appropriate checklist; use section 4 to develop local guidelines for in-house documentation.
3. Begin the evaluation and documentation of existing programs.
4. Use section 2 to seek new programs.
5. Read section 6.

Tour Group E. *Our members are interested in selling or exchanging computerized instructional materials. Each of us has programming experience, and a broad range of programming skills is represented within the group. Some of us devote considerable time and energy to producing programs that we hope will be used by many teachers; others of us are concerned primarily with improving the quality of those programs or facilitating their exchange.*

Itinerary:

1. Read sections 1 through 6.
2. Working with experienced users of computerized instructional materials, use sections 3 through 5 to develop appropriate instruments and procedures for evaluating and documenting programs.
3. Insist on the use of those instruments and procedures to guide the writing or exchange of computerized materials.

Please join your tour group now. Should you choose to deviate from its proposed itinerary, the tour guide cannot accept responsibility for your safety or for any professional expenses that you may incur.

Have a pleasant journey.

SECTION 1

What, Why, and for Whom?

Let the Purchaser Beware!

The definitions, instruments, and procedures that compose the *Guidelines* are intended to be practical resources for persons involved with the selection or creation of computer programs to be used for instructional purposes. The evaluative issues that may be inferred from the instruments of the *Guidelines* (and the directions for their use) represent state-of-the-art knowledge, experience, and intuition regarding what constitutes a "good" computer program.

A cursory reading of this document may very well produce the feeling that this booklet raises more questions about evaluating instructional computer programs than any reasonable person would ever want answered. Indeed it does. Evaluative criteria in this area are both numerous and tentative. Many are expressions of group or personal values, thus being judged to be of utmost relevance by some evaluators while being viewed as totally irrelevant by others. These materials were designed to bring a wide range of evaluative considerations to the reader's attention in a format that would permit easy adaptation to meet the user's priorities. Users of these *Guidelines* should recognize, therefore, two assumptions underlying their development. First, it is highly likely that a given user will assign greater weight to some criteria than to others and will judge some criteria to be of no importance at all. Second, it is the proper and necessary task of the user to select and weigh the criteria to be employed in the local evaluation of computer programs.

The role of any set of guidelines is to *guide* the educational decision maker, supporting the careful and efficient review of an issue; these *Guidelines* are no exception. They do not attempt to employ numerical ratings to generate purchase-rejection decisions. When adapted to reflect a given user's selection of criteria, however, they will focus on the issues critical to those decisions.

Something Old, Something New

Almost every educator is eventually called on to select learning materials to be used not only in one's own classroom but also in those of many colleagues. Such selection will generally be guided by written criteria or checklists that attempt to integrate national and local professional opinions.

Most commonly, selection tasks have focused on printed instructional resources, both basic texts and supplementary materials. Selection criteria have included concerns about the scope and sequence of topics covered by the resource, instructional approaches employed, the type of student activities involved, readability, societal issues (e.g., sex or racial discrimination), appearance, durability, cost, and the relative ease with which the product can be used by both teacher and student. The order of importance of the criteria might differ from situation to situation, but the issues are familiar.

The need to evaluate computerized instructional materials may exceed that of printed materials, since computerized lessons are often executed relatively independently of teacher or user control. For example, the use of a set of textbook activities is subject to teacher control through selection, modification, augmentation, and suggested ordering. The student exercises additional control of the assignment in terms of pacing, reordering, and selective deletion. Such manipulation may be difficult (or, perhaps, impossible) in a computerized lesson.

Companies, organizations, and individuals are now engaged in the production of computer programs

aimed at the educational market. The majority are intended to fulfill drill, practice, instructional-management, or limited tutorial functions and deal with isolated concepts. As the number, variety, and complexity of programs increase, we approach that desirable point at which we will be faced with choosing from among several approaches to a given teaching situation. Unfortunately, the old procedures for sampling and selection will no longer be adequate. The guidelines presented in this booklet are a response to the need for standards for evaluating the unique instructional materials made possible by computers.

Some Words about Words

The growth of instructional computing has been accompanied by the generation of considerable terminology—often the use of existing words in new and precise ways. Seven terms central to the use of these materials are defined operationally below.

Hardware consists of an individual item (or combination of items) of mechanical or electronic gadgetry required for the execution of a computer program. In addition to the computer itself, examples of common hardware items include monitors, printers, card readers, and cassette tape recorders.

Software refers to a computer program in a form that can be executed using equipment available to the evaluator. The term also includes any materials exterior to the program but referenced by it or needed to explain the instructional use of the program. (Although this inclusion represents a departure from standard usage of the term, the extended definition suits our purposes here.) These exterior materials are often called *ancillary materials* and include such things as tables, charts, laboratory equipment, lexicons, workbooks, and directions regarding appropriate uses of the program.

Software clusters are collections, usually sequenced, of interrelated programs. The procedures for evaluating such clusters give rise to sampling issues that do not apply to the evaluation of an isolated, stand-alone item of software.

Courseware is a set of computer programs with which a student has direct interaction during a learning sequence. The use of the term *courseware* is much more restrictive than *software* as used in these *Guidelines*, since software also refers to management or utility programs with which the learner has no direct interaction. For us, then, courseware is a subset of software. (Many persons restrict the use of the term *software* to computer programs that are "invisible" to the learner; the authors chose a more inclusive definition of the term.)

Documentation includes all printed materials that describe either the computer software (e.g., its purpose, cost, intended audience, effectiveness) or the equipment required for the use of that software.

Systems-error messages are statements that are generated by the computer during the execution of a program but that are not a part of the instructional computer program itself. They arise as the result of some error in the instructional program (often a result of sloppy or lazy programming) or the peculiarity of the computer system being used to execute the program. At best, such statements distress and confuse the person executing the program, since they are usually written in a kind of "computerese" that is meaningful only to a programmer. At worst, they terminate the execution of the program. Consider the plight of a teacher who, having spent several minutes typing a text passage for analysis by a computerized readability program, accidentally types "YAS" rather than "YES" to some option regarding that analysis and receives the following error message:

INPUT ERROR

READY

There will be no small amount of frustration when it is learned that the passage has been "lost" and the project must be redone.

Program listings are printed forms of computer programs. Written in the language of a particular computer, they can be entered (typed) into *that specific kind of computer* by a careful typist who has had a few hours' instruction. To be of use with a *different kind of computer*, the program possibly would need to be rewritten by a computer programmer.

Looking Ahead

The use of computerized instructional material involves five principal tasks: (1) obtaining software and its documentation, (2) evaluating the materials obtained, (3) making a decision regarding the acquisition of the software, (4) establishing a system for helping all potential users to locate programs of instructional interest within the bank of programs on hand, and (5) modifying or improving the programs in the bank. These tasks must be addressed not only for purchased software but also for locally developed software and for software obtained at little or no cost from a nonprofit software exchange group.

The first four tasks can be carried out by any educator interested in making instructional use of computerized materials; they are the subject of sections 2 through 4 of these *Guidelines*. The fifth task requires an understanding of the topic addressed by a given program, general knowledge about computers, and the ability to write programs in the given computer language. Addressed by section 5, this task may require a team effort in which one team member is an experienced programmer. Section 6 goes beyond the primary goals of the *Guidelines* by providing a limited bibliography for readers interested in exploring the topics of computer literacy, the social impact of computers, the teaching of computer programming, and the use of computers in mathematical problem solving.

SECTION 2

Obtaining Software and Documentation

What's Out There?

There are several kinds of sources of information about commercial and noncommercial computerized instructional materials. Descriptions, sources, and reviews of commercial programs are found in a number of periodicals (see References, part 1 at the end of this section). These publications often contain program listings and descriptions of programs offered for use without further charge. In addition to these periodicals directed toward computer novices, there are works for those who have programming experience or technical knowledge of microcomputers (part 2).

For the person wishing to survey the field of computing journals in search of appropriate resources, three publications will have special appeal (part 3). The first of these (3a) is a where-to-find-it guide to persons, periodicals, and projects in instructional computing; included are annotated bibliographies to a variety of publications in the field. The second item (3b) is a quarterly subject index (with annotations) to the articles and software reviews of approximately forty computing journals. The final reference (3c), comprising reprints of the tables of contents of the most recent issues of about seventy computing journals, includes a subject index.

Publications devoted primarily to the issue of computer software can be placed in one of two categories: software directories or software reviews. Software directories (part 4) provide basic information (name, source, topic, price) for a vast bank of software available from a wide range of producers. Some directories are indexed by subject, grade level, microcomputing system, or publisher; a few include some evaluative comments. Most directories include software for several microcomputer systems, but others are specific to Apple (4a, b, d, g, m, o, q), Commodore (4c), IBM (4h), Radio Shack (4i), or Texas Instruments (4p) systems. Software review publications (part 5) go beyond the directories restricted to reporting factual information. Using evaluative criteria and procedures, these sources offer opinions regarding the technical and educational quality of the software referenced. Although "professional" software reviews (including those appearing in the journals of educational organizations) are convenient, the convenience is not without a price. In accepting the conclusions of the reviewers, we also accept the

(often unstated) criteria and procedures of the review process—not to mention the personal interests and prejudices of the reviewers. Truly, there is no such thing as a free lunch.

Software exchanges are organizations that provide a mechanism for the free or inexpensive exchange of computer software among a large number of users. At present, the majority of such exchanges are "software jungles" that users enter at their own risk. Generally speaking, participation in the exchange neither requires nor implies any particular standards of program quality. Because such exchanges are operated with little or no funding, their maintenance often depends on the freewill contributions of a handful of dedicated professionals. Local software exchanges of this type are numerous; computer users in your area probably know of several. A partial listing can be found in reference 3a.

In many metropolitan areas, there are educational stores acting as software vendors. They carry a great deal of the educational software, and they will obtain software for districts to preview. These vendors may be media stores, educational supplements houses, or stores that handle educational toys.

National centers for the cataloging, evaluation, exchange, or sale of software are in the formative stage (References, part 6). When fully operational, they will offer a variety of services through a variety of means. The development of such centers is funded both publicly and privately; their maintenance will probably be through the fees and subscriptions of users.

Finally, private and governmental organizations are supporting the development of exemplary computerized materials and model plans for the use of computers in instruction. One can obtain a list of those projects funded by the National Science Foundation (NSF) or the National Institute of Education (NIE) by contacting NSF (1800 G Street, NW, Washington, DC 20550).

Into the Jungle: Two Scenarios

Given the ever-growing pool of software clusters and isolated items of software, persons interested in the use of computerized instructional materials can use the sources of information just noted to identify programs of potential value. Let's assume that you've accomplished that much—that is, you find yourself in one of the following situations:

Scenario 1: You have the name and source of a given item of software or a software cluster but do not have the program in a form ready to execute on available hardware.

Scenario 2: You are in possession of one or more items of software that can be executed using available hardware.

If scenario 2 fits your situation, you may wish to go directly to section 3. If not, let's turn to the task of obtaining documented software for evaluative purposes.

Getting the Stories Behind the Names

So you've got the names of what appear to be some promising items of software. The next task is to obtain copies of that software, right? Well, maybe yes and maybe no. Are you certain that it was designed to run on the kind of hardware that you have available? Does your computer have a large enough memory to execute the program? Does the use of the program assume the availability of certain kinds of laboratory equipment, reference books, or workbooks? How much will it cost to obtain the software for review purposes? If the producer of the software will not release it for free review, is the software in use somewhere nearby? If the object of interest is a large software cluster, is it possible to obtain or even purchase selected programs from the cluster?

You may decide, depending on the answers to such questions, to drop the idea of obtaining the program (e.g., it can't be executed using the hardware available to you). Since the viewing of the software may involve paying for all or part of it, there is good reason to seek some basic descriptive information before trying to secure the software itself. The form letter on page 9 gives examples of the questions you can and should expect a software supplier to be able to answer, either directly or through promotional literature.

In listing the programs of specific interest in this letter, you will need to take special care in selecting sample programs from a software cluster. If the cluster is large, it's doubtful that the producer will be willing to make all of it available for review. Nevertheless, you want the sample from the cluster to include

programs that will reveal the extent of interdependence of programs within the cluster. Your selections, therefore, probably should include the following:

- A program near the beginning of the cluster. This may reveal programmer/user agreements (conventions) used throughout the cluster.
- At least two programs that form a sequence within the cluster. This sequence can suggest the degree to which review is built into the cluster. It may reveal that the programs are interdependent to such a degree that a purchase of parts of the cluster may not be educationally feasible.
- A program near the end of the cluster. This program may give you some idea of the upper limit of the material covered by the cluster. It also may show the extent to which the style of presentation has changed or matured.

In 1981, some publishing companies were predicting that computerized instructional materials would soon account for one-third of their educational sales. That kind of market size, in turn, guarantees that brochures containing the information requested by our form letter will be available and that procedures for reviewing the materials will be developed.

The Wheat (Maybe) and the Chaff

Some items of software may be eliminated from further consideration, based on the descriptive information obtained from the vendor, by failing to meet a critical condition of hardware compatibility, availability for review, price, and so on. Weak documentation by the vendor may make some software "iffy"; other items will show real promise. Clearly, such classification is a function both of the quality of the documentation and of the standards of the user.

Having searched the literature to obtain answers to the questions raised in the form letter, the careful reviewer would retain that information on the front page of a software documentation sheet like the one on page 19. If and when the software becomes part of a school system's instructional resources, such documentation will be invaluable to potential users other than the reviewer.

REFERENCES

A brief guide to these references can be found beginning on page 5.

1. For the Computer Novice

- a. *Creative Computing* (P.O. Box 789-M, Morristown, NJ 07960).
- b. *Classroom Computer Learning* (P.O. Box 266, Cambridge, MA 02138).
- c. *The Computing Teacher* (Computing Center, Eastern Oregon State University, LaGrande, OR 97850).
- d. *Educational Computer Magazine* (P.O. Box 535, Cupertino, CA 95015).
- e. *Electronic Education* (Suite 220, 1311 Executive Center Drive, Tallahassee, FL 32301).
- f. *Electronic Learning* (Scholastic, Inc., 50 West 44th Street, New York, NY 10036).
- g. *InfoWorld* (375 Cochituate Road, Route 30, Framingham, MA 01701).
- h. *Journal of Computers in Mathematics and Science Teaching* (Box 4455, Austin, TX 78765).
- i. *School Microcomputer Bulletin* (Box 1326, Holmes Beach, FL 33509).

2. For Those with Programming Experience or Technical Knowledge

- a. *BYTE—Small Systems Journal* (70 Main Street, Peterborough, NH 03458).
- b. *Compute!* (515 Abbott Drive, Broomall, PA 19008).
- c. *Microcomputing* (P.O. Box 997, Farmingdale, NY 11737).
- d. *Micro—The 6502/6809 Journal* (P.O. Box 6502, Chelmsford, MA 01824).

3. Computing Journals with Lists of Resources

- a. *Classroom Computer Learning Directory of Educational Computing Resources* (Intentional Educations, 341 Mount Auburn Street, Watertown, MA 02172).
- b. *Microcomputer Index* (Microcomputer Information Services, 2464 El Camino Real, Suite 247, Santa Clara, CA 95051).
- c. *Micro . . . Publications in Review* (Vogeler Publishing, P.O. Box 489, Arlington Heights, IL 60006).

4. Software Directories

- a. *The Addison-Wesley Book of Apple Computer Software* (The Book Company, 16720 Hawthorne Boulevard, Lawndale, CA 90260).
- b. *The Apple Software Directory: Education* (WIDL Video, 5245 West Diversey Avenue, Chicago, IL 60639).
- c. *Commodore Software Encyclopedia* (System Marketing Group, 681 Moore Road, King of Prussia, PA 19406).
- d. *Educator's Handbook and Software Directory* (Vital Information, 7899 Mastin Drive, Overland Park, KS 66204).
- e. *Index to Computer-based Learning* (Instructional Media Laboratory, University of Wisconsin, P.O. Box 413, Milwaukee, WI 53201).
- f. *Instructor's Computer Directory for Schools* (P.O. Box 6099, Duluth, MN 55806).
- g. *Micros in Mathematics Quarterly* (Microcomputer Curriculum Project, Price Laboratory School, University of Northern Iowa, Cedar Falls, IA 50613).
- h. *PC Clearinghouse Software Directory* (PC Clearinghouse, 11781 Lee-Jackson Highway, Fairfax, VA 22033).
- i. *Pipeline* (P.O. Box 388, Iowa City, IA 52244).
- j. *Radio Shack TRS-80 Educational Software Sourcebook* (available from any Radio Shack dealer).
- k. *Reference Manual for the Instructional Use of Microcomputers* (JEM Research, Discovery Park, P.O. Box 1700, Victoria, BC, V8W 2Y2, Canada).
- l. *School Microware Directory* (Dresden Associates, P.O. Box 246, Dresden, ME 04342).
- m. *Skarbek Software Directory* (11990 Dorsett Road, Saint Louis, MO 63043).
- n. *Sources for Courses* (TALMIS, 115 North Oak Park Avenue, Oak Park, IL 60301).
- o. *Swift's Software Directory: Apple II Edition* (Sterling Swift Publishing Company, 1600 Fortview Road, Austin, TX 78704).
- p. *Texas Instruments Home Computer Program Library* (available from any Texas Instruments dealer).
- q. *VanLoves Apple II/III Software Directory* (Vital Information, 7899 Mastin Drive, Overland Park, KS 66204).

5. Software Review

- a. *Computer Program Reviews* (P.O. Box 331, North Springfield, VT 05150).
- b. *Courseware Report Card: Elementary (K-6) Edition* (150 West Carob Street, Compton, CA 90220).
- c. *Courseware Report Card: Secondary (7-12) Edition* (150 West Carob Street, Compton, CA 90220).
- d. *InfoWorld Report Card on Microcomputer Software* (375 Cochituate Road, Box 837, Framingham, MA 01701).
- e. *Journal of Courseware Review* (Apple Education Foundation, 20525 Mariani Avenue, Cupertino, CA 95014).
- f. *MACUL Journal* (% Lary Smith, Wayne County Independent School District, P.O. Box 807, Wayne, MI 48184).
- g. *School Microware Reviews* (Dresden Associates, P.O. Box 246, Dresden, ME 04342).
- h. *Software Review* (Meckler Publishing, 520 Riverside Avenue, Westport, CT 06880).
- i. *TALMIS Courseware Ratings* (115 North Oak Park Avenue, Oak Park, IL 60301).

6. National Centers for Cataloging Evaluation, Exchange, or Sale of Software

- a. CONDUIT, P.O. Box 388, Iowa City, IA 52244.
- b. EPIE Institute, Box 839, Water Mill, NY 11976.
- c. Microcomputer Education Laboratory, QAL/MEL, 217 Baldy Hall, SUNY at Buffalo, Amherst, NY 14260.
- d. MicroSIFT, Northwest Regional Educational Laboratory, 300 SW 6th Avenue, Portland, OR 97204.
- e. TABS Project, Arps Hall 202-A, 1948 North High Street, Columbus, OH 43210.
- f. Technical Education Research Centers (TERC), Computer Resource Center, 8 Eliot Street, Cambridge, MA 02138.

SAMPLE REQUEST-FOR- INFORMATION FORM LETTER

Dear _____:

We wish to evaluate certain of your computer programs for possible purchase. This evaluation would include any related teacher- or student-support materials. The following programs are of specific interest:

_____	_____
_____	_____
_____	_____

Before requesting the programs themselves, we need information (brochures, technical reports, or your comments) to answer the following questions:

1. What are the system and memory requirements for each program's use?
2. On what media (disks, cassette tape, cartridge, etc.) is each program available?
3. In what computer language(s) is each program written?
4. For which grade levels or ability levels is each program intended?
5. What are the specific content areas or instructional objectives for each program?
6. Has each program been tested or validated in a classroom setting?
7. What kind of instructional materials are available to support a given program's use? What is the cost of such materials?
8. What is the procedure or cost for obtaining each program for a full review?
9. What is the purchase price of each program?

Please send this information to

Thank you for your assistance.

Sincerely,

SECTION 3

Getting Hardnosed about Software: Guidelines for Software Review

Getting the Message Out of the Medium

Once basic documentation and a program for review have been obtained, the important process of evaluation can begin. The computer program is evaluated first, followed by a review of the related documentation and ancillary materials. The evaluation serves only to provide information on which a decision can be reached; it does not use weighted scales or numerical totals to make the decision for you.

The evaluation model presented in this section is both comprehensive and detailed. It should be tailored to meet the specific needs of a given user. In doing such tailoring, however, the user should be careful to ensure that the resulting version of the model considers each of seven important steps of the review process.

Step 1. Load the program on your system. Run the program briefly to become familiar with the program's "flow."

Step 2. Execute the program as a successful student would, avoiding intentional or careless errors. Extend the program when possible by interacting "creatively" as a good student would in testing the cleverness of the programmer.

Step 3. Execute the program as an unsuccessful student would. Respond incorrectly to test how the program handles student errors. If an erroneous response to a task results in the repetition of that task, make repeated incorrect responses. Be sure to repeat the *same* response and also try giving *different* incorrect responses. Finally, make other kinds of errors such as typing mistakes, incorrect form of input (e.g., "one" for "1"), content errors, and errors in following the directions.

Step 4. Observe and take notes while the program is executed by two or three students from the group with which it is to be used. Do not offer help or ask questions during their interaction with the program; assist only when a student's progress seems totally blocked. As soon as a student completes the activity, conduct an interview to get his or her general impression of the program as well as answer questions about points of difficulty.

Step 5. Use the checklist for the evaluation of software on pages 13-14, adapting it as necessary to fit your own needs. Throughout, your responses should be based *only* on your observations and experiences with the program. Once completed, your responses can be compared with the vendor's claims.

Step 6. Complete the checklist for the evaluation of documentation and ancillary materials. Again, the extent of the review will depend on your needs and the impressions remaining from your review of the actual software.

Step 7. Make a decision.

The design of the sample Software Evaluation Checklist calls for items on one side of a sheet of paper and abbreviated explanations of item vocabulary or purpose on the other side of the sheet. (Detailed discussions of the terms used on this sample are presented in the next subsection of these *Guidelines*.) This one-sheet design is judged highly desirable insofar as it simplifies paper shuffling. (Of course, the items and their explanatory notes could be placed on your computer. This not only saves trees and storage space but also illustrates that you recognize another obvious advantage of computers in education.)

Although the sample checklist here purposely includes an excess of items, in practice a shorter form

may do the job. Moreover, experienced evaluators would require few, if any, of the explanatory notes given on the back of the sheet.

Words to Evaluate By

The use of our checklist, or any other checklist, by more than one person requires an agreement regarding the meaning of terms used to classify or evaluate programs. We must extend the minilexicon we started in section 2. We'll do this by proceeding item by item through the sample checklist, stopping to define and discuss concepts as the need arises.

1. Instructional Range

For any given program, the appropriate grade level and ability level must be judged according to the particular instructional setting. (A program that is informational at first grade might be remedial at third.) Key factors in determining grade or ability level are the concepts in the program. Other important factors are reading level, prerequisite skills, degree of student control, and intended instructional use. It is possible for a program to be flexible enough to be used across a wide range of grade levels and ability levels.

2. Instructional Grouping

The interaction between the learners and a program can be either direct or through an intermediary (the teacher). The first two groupings described below involve direct interaction; the third uses an intermediary.

Individual. Some programs are designed specifically for use by one person. Programs of this nature may include a summary of individual performance; others may allow a specific level of difficulty—a specific set of spelling words, for example. Programs for individuals may have provisions for easier or more difficult situations, depending on the individual student's response.

Small group. Some programs are specifically designed for use with two, three, or four students. One such example might be an arithmetic fact game involving a car race in which each student can move forward on a correct answer. Other programs for small groups might be business simulations in which each small group is a business.

Large group. Large-group use of programs may vary in intent. A program may be used with a large group solely for the purpose of introducing the program for later use by individuals or small groups. Some programs are designed for use by a teacher in graphically demonstrating a concept. (Trigonometric functions provide one example; fraction concepts such as equivalence are another.)

3. Execution Time

The time required for the use of a program will vary considerably. The load time of the program depends on the complexity of the program and on whether it is being loaded from a cassette tape or a disk. (The disk is a much faster but more expensive storage medium.) The type of program is a factor (e.g., tutorial versus informational). The user will require additional time if responses are consistently incorrect. Another time factor is the degree of user control (i.e., how many options are available to the user). An estimate given as a range is an appropriate response for this item.

4. Program Uses

The classification of programs by the instructional uses to which they are put is difficult. Careful analysis notwithstanding, definitions seem doomed to fuzziness, and a given program may seem to fit in more than one category. Moreover, the use of a given program can often be determined by the classroom practitioner; a program written with one use in mind may be used in completely different ways by each of several persons. In fact, a given person may identify more than one use for a program, leading to the necessity to indicate *primary* and *secondary* uses.

Despite the difficulties just noted, there is instructional value in attempting to identify the use or uses

of a program. The following definitions are proffered as an aid to the classification of those program uses that appear on the checklist.

Demonstration: Programs that assume the intervention of an informed manager (teacher) between the program and the learner. They are designed to permit the manager to pursue the questions and suggestions of students by varying the conditions associated with some concept being developed. These programs are generally devoid of textual display relative to their purpose and operation. This lack of text, while precluding direct use of the program by the learner, permits efficient realization of the program's purpose by an informed user (teacher). External documentation is critical to demonstration programs.

Drill or practice: Programs that assume previous instruction in the concept, skill, or process to be addressed. They present a controlled sequence of exercises designed to drill the recall of certain pairings or to practice some algorithm.

Informational: Programs designed to generate data, with no suggestion as to their use. They offer very little interaction with the person for whom the data is generated.

Instructional gaming: Programs calling on the user to apply one or more specific skills or concepts in a game environment. The game dimension of the environment includes the conditions of competition with oneself or others, specific (if arbitrary) rules, the need to develop a winning strategy, and the introduction of random events to force the revision of strategies.

Instructional management: Programs designed to aid the user in the flexible and efficient management of such limited instructional resources as time, equipment, workspace, instructional personnel, and consumable materials. Such programs assume, at least in part, the tasks of making instructional diagnoses, giving course assignments, evaluating student progress, keeping student records, and facilitating communication among instructors and students.

Instructional support: Programs designed to facilitate the generation of instructional products or materials, the location of instructional resources, or the selection of instructional products.

Problem solving: Programs serving either the student or the instructor. The problem to be investigated may be either within a given subject or within instruction in this subject. In practice, there are two levels of use—one that employs existing programs using known algorithms and one that requires the development of a program employing a user-defined algorithm. In fact, the development of the algorithm itself may be the object of the problem-solving activity.

Programming utility: Program building blocks (or complete lesson authoring systems) that may be used to construct instructional programs. Their availability allows instructors or programmers to make efficient use of their hardware and their programming time.

Simulation: Programs that attempt to represent key aspects of some environment within which the user will need to make decisions and will be informed of the results of those decisions *without experiencing the real consequences of possible misjudgments*. The time required to develop and use simulations with high fidelity is justified in situations where actual experience is ruled out because of expense, safety considerations, or the time required for the actual experience.

Testing operations: Programs to aid in criterion-referenced testing or tailored testing. These programs offer such services as the banking of objectives or test items, item analysis, record keeping, test generation, and test scoring.

Tutorial: Programs through which the computer assumes total responsibility for instruction. Tutorial programs are characterized by a dialogue between the student and the computer in which the direction and level of the dialogue are shaped by student input. Ideally, such programs would be high-fidelity simulations of the best teaching behavior associated with a given topic and would control the variables associated with the what, why, when, and who of the instructional episode.

Whistles and bells: Programs designed to acquaint the user with those characteristics of a computer system that may have instructional application.

Word processing: Programs that can be used to create, store, modify, and print the text (and, perhaps, simple graphics) of letters, reports, topical papers, and standard forms. They are used to support the improvement of written materials produced by teachers and students.



1. INSTRUCTIONAL RANGE

2. INSTRUCTIONAL GROUPING FOR PROGRAM USE

3. EXECUTION TIME

4. PROGRAM USE(S)

5. USER ORIENTATION: INSTRUCTOR'S POINT OF VIEW

6. USER ORIENTATION: STUDENT'S POINT OF VIEW

7. CONTENT

8. MOTIVATION AND INSTRUCTIONAL STYLE

low high

* * * * *

degree of student control

9. SOCIAL CHARACTERISTICS

<i>present and negative</i>	<i>not present</i>	<i>present and positive</i>	
_____	_____	_____	competition
_____	_____	_____	cooperation
_____	_____	_____	humanizing of computer
_____	_____	_____	moral issues or value judgments
_____	_____	_____	summary of student performance

Key to Terms: Software Evaluation Checklist

1. The grade levels and ability levels for a particular program are primarily determined by the concepts involved. Other important factors are reading level, prerequisite skills, degree of student control, and intended instructional use. It is possible for a program to be flexible enough to be used across a wide range of grade levels and ability levels.

2. Some programs are designed for use by individuals. Others have been or can be modified for participation by two or three persons at a time. Simulations or demonstrations often pose opportunities for large-group interaction. A given program may be used in more than one grouping, depending on the instructor.

3. The time required for the use of a program will vary considerably. Include loading time for cassettes. A time range is the appropriate response here.

4. Instructional programs can be categorized according to their uses. Some programs may have more than one use, thus falling into more than one of the following categories:

Demonstration: Used by a teacher for leading group discussion.

Drill or practice: Assumes that the concept or skill has been taught, merely reinforces.

Informational: Generates data but does not direct the use of those data.

Instructional gaming: Focuses on decision making within a game format; involves random events and the pursuit of a winning strategy.

Instructional management: Helps manage the learning environment by assuming major responsibility with respect to student assessment, record keeping, and generating learning prescriptions.

Instructional support: Generates instructional materials, helps locate instructional resources, or aids in selecting instructional products.

Problem solving: An algorithm that serves the problem solver during the exploration of a complex problem.

Programming utility: Programs that help write other instructional programs.

Simulation: An electronic slice of life.

Testing operations: Provides services such as item banking, test generation, test scoring, and item analysis.

Tutorial: A dialogue between student and computer on a topic that is not yet understood by the student.

Whistles and bells: Demonstrates features of the computer system—features that may be of value in instructional applications.

Word processing: Permits the efficient improvement of written materials.

5. These are factors relevant to the actual use of the program from the point of view of an instructor.

Flexibility: A program may allow the user or the instructor to adjust the program to different ability levels, degrees of difficulty, or concepts.

Intervention or assistance: A rating of "low" means considerable teacher intervention or assistance is required.

6. These are factors relevant to the actual use of the program from the point of view of a student.

Directions: The directions should be complete, readable, under the user's control (e.g., should not scroll off the screen until understood), and use appropriate examples.

Output: Program responses should be readable, understandable, and complete. If in response to student input, the output should be of an acceptable tone and consistent with the input request.

Screen formatting: The formats during a program run should not be distracting or cluttered. Labels and symbols should be meaningful within the given context.

External information: A program may require the user to have access to information other than that provided within it. This may include prerequisite content knowledge or knowledge of conventions used by the program designer as well as maps, books, models, and so on.

System errors: System errors result in the involuntary termination of the program.

Input: The user should know when, where, and in what form input is expected. Student input rather than automatic timing should prompt the presentation of new material on the screen, and specific screen areas should be reserved for such input. Characters with special meanings should be avoided; however, when they are needed, such symbols should be accompanied by explicit instruction and by a readily available "help" display regarding their use. The complexity of typed input should be minimized but should be sufficient to serve the learning objectives of the program.

7. These are matters relevant to the subject-matter content of the program.

Focus: The program topic should be clearly defined and of a scope that permits thorough treatment.

Significance: The instructional objectives of the program must be viewed as important by the instructor. Also, the program should represent a valid use of the computer's capabilities while improving the instructional process.

Soundness or validity: The concepts and terms employed should be correct, clear, and precise. Other important factors are the rate of presentation, degree of difficulty, and internal consistency.

Compatibility: The content, terminology, teaching style, and educational philosophy of the program should not conflict with those generally encountered by the student.

8. These items are self-explanatory.

9. Competition, cooperation, and values are concerns that may be a function of the way a program expresses them. (War gaming and the "hangman" format are sample issues.) Also, the "humanizing" of the computer may serve for motivation or to reduce anxiety, but it also may become tedious, misleading, and counterproductive.

The summary of student performance can be dichotomous (win or lose), statistical (time expended or percent of items correct), or subjective (as in the evaluation of a simulation). It may be for student, teacher, or both.

5. User Orientation: Instructor's Point of View

Flexibility. A program may permit the user or instructor to adjust it to accommodate a range of ability levels, several degrees of difficulty with respect to a given user, or the class of concepts involved. This can be accomplished through options presented by the program. For instance, a program dealing with the practice of whole number computation might offer a choice of the operation (addition, subtraction, multiplication, division) or the range of numbers involved (0-10, 0-20, 0-100). Another option might involve the difficulty of the problems presented in terms of the procedure required for a solution (i.e., borrowing versus no borrowing in the subtraction problems). Programs that do not offer such options to the student may have a provision for the execution of such adjustments by the teacher.

Intervention or assistance. Although it is a desirable goal for most programs to operate independently of the classroom teacher, there are programs for which assistance is needed and even desirable. For instance, some drill programs are "unending." Intervention to stop the program is done by the teacher and involves a judgment of what constitutes sufficient program use by a given student. (Note: A rating of "low" on "freedom from need to intervene or assist" would mean that considerable teacher intervention or assistance is required.)

6. User Orientation: Student's Point of View

Quality of directions. Directions need to be concise and clear. They should be developed for the lowest level of use. Some directions can be simplified by the use of an example.

Quality of response. Responses to student input should be low-key and understandable. Responses to incorrect input should be neutral (e.g., "Incorrect. Try again." versus "Wrong, dummy! How could you get this far and know so little?").

Screen formatting. Formatting refers to the physical layout of text and graphics presented on the screen. Examples of poor formatting include a full screen of text with single spacing, too many graphics, continually flashing displays, and text scrolling off the screen. Some good examples are centering of data, a few items on the screen with the user in control of when the next items appear, and reverse-field printing of key items (i.e., light print against a dark screen versus dark print against a light screen).

Need for external information. A program may require the user to have access to information other than that printed by it. Generally speaking, the more self-contained a program is, the better. However, certain programs intentionally require the use of external items. Examples of such programs include programs dealing with dictionary skills and the use of maps.

Disruption by system error. System errors result in the involuntary termination of a program. They may be introduced by improper calculations, errors in the logic of the program, or input that is conceptually correct but does not meet the form required by the program. Programs written to prevent all systems errors—syntactic errors and improper input—are the most desirable. They can be developed by using special input routines.

Simplicity of student input. The program should ensure that a user knows when, where, and in what form input is expected. Generally speaking, new displays should be initiated by student input rather than an automatic timing loop. Input locations should be standardized throughout the program. Characters with special meanings (such as the "*" to return to the program's main menu) should be avoided wherever possible in favor of clear, descriptive input (e.g., typing "MENU" to return to the program's main menu). If special symbols are necessary, explicit instruction in their use—including appropriate examples—should be given; thereafter, a "help" display reminding the user of how to employ those symbols should be readily available. And although complex input procedures should be kept to a minimum, it is important to note that such procedures may be justified. For instance, if the form of the input is itself a focal issue in the learning activity (e.g., the proper use of diacritical marks in a foreign language program or exponents and Greek letters in the statement of mathematical formulas), then the learning objective should not be sacrificed to the cause of brief, simple input.

7. Content

Instructional focus and significance. The program's topic should be clearly defined. The instructional objectives of the program must be viewed as important by the instructor. Also, the program should

represent a valid use of the computer's capabilities. (The program should not repeat the same set of questions or present problems in exactly one sequence.)

Validity and compatibility. The concepts and terms used should be correct, clear, and precise. The rate of presentation should be consistent with the levels for which the program is intended. The content, terminology, teaching style, and educational philosophy should not conflict with those generally encountered by the student.

8. Motivation and Instructional Style

There are many options that may increase student motivation or develop a particular instructional style. Some of these options are a function of the hardware being used. Because these features are optional, a column labeled "none," meaning the feature is not present, has been added at the left of the low-high continuum.

Student involvement. The proper degree of involvement will vary with the type of program used. A drill or practice program for one student is highly active, whereas an informational program may require little student input. Tutorial programs, being dialogues, should require student involvement.

Student control. Several concepts may be involved here. Options available to the student at the beginning of the program may include different kinds of content, levels of difficulty, or the type of response the student may wish to use. Additional examples of student control include the selection of the number of problems to be done and the provision of help (at the student's request) at any point in the program.

Game format. The use of a game format often generates favorable responses from students. Some examples are mathematics drill as a drag race or in a tic-tac-toe or concentration format. Caution must be exercised so that the game format does not overshadow the instruction. Note that a game format does not imply that the program use is instructional gaming; a drill program may use a game format. A focus on decision-making strategies is required for a program to be classified as having an instructional gaming use.

Graphics, animation, color. The functional use of one or more of these features can enhance instruction; their improper or irrelevant application can interfere or cause misconceptions. The uses of graphics, animation, or color must be examined carefully to ensure that they support concept development. One might begin that examination by asking what would be lost if the feature were deleted.

Voice and nonvoice audio. The use of voice input and output has considerable potential. Be aware that the quality and use of voice is dependent on the hardware. The use of nonvoice audio provides an additional means for attracting interest but may be distracting in the classroom.

Light pen. Some programs require the use of a light pen. This can be a good feature if a light pen is available, but a program employing a light pen will be useless without it.

Ancillary materials. Materials external to the program may be used. Some may be provided, but their use may not be required (e.g., pretests or posttests, worksheets, etc.). Others may be necessary but are locally available (e.g., dictionaries or local road maps). Care should be taken to note any materials developed especially for the program that must be supplied by the program's author or publisher.

As in item 4, a blank is left at the end of item 8. This is necessary because the list of special features grows daily; its length is a function both of the subject matter and the computer system involved. The listed features are in relatively common use; other examples include touch panels (the user can execute an instructional task simply by touching the screen with a finger), special keyboards for the physically handicapped, electronic devices that monitor a physical phenomenon (such as water temperature) and use it in the program's execution, and mechanical devices (robots, if you will) that interact with the program.

9. Social Characteristics

Competition and cooperation. Some programs may be written for use by a single student, several students, or the whole class. Competition may be between two or more students or between a student and the computer; it is generally motivating. The need for cooperation among students, also desirable, might be found in a business simulation that requires several students to operate the business.

Humanizing the computer. Programs may give the computer human characteristics. Some people find this desirable, whereas others wish to emphasize that the computer is a machine. Asking for the student's name may or may not be necessary or desirable. Older students generally find it boring, distracting, or phony.

Moral issues. Items or techniques that at first create interest may be questionable from other points of view. Capital punishment as exhibited by most hangman games is one example; the violence of war games is another.

Summary of student performance. A performance summary is a feature inherent to games that result in a winner. Performance on a simulation (e.g., the use of a credit card) also calls for a summary evaluation (e.g., the credit card was cancelled because of the inability to meet payments regularly). Drill programs often conclude with a summary regarding the speed or accuracy of the student's performance. Depending on the manner in which these summaries are presented, they may have either a positive or a negative effect on the learner.

Notes on the Design or Redesign of the Checklist

With the exception of the few descriptive requests at the top of the form and in items 1 through 4, the use of the checklist requires only making checks, circling words, or bracketing ranges of dots on judgment scales. The design of the scales in items 5 through 9 was with an eye to consistency; it also consistently expresses the summary prejudices of the writers. For each scale, a rating toward the left-most extreme (identified by such terms as "low," "passive," and "poor") would generally be, for the writers, a negative evaluation of the software. (This technique isn't foolproof. A rating of "low" on "freedom from need for external information" would be undesirable for a drill program but might be necessary—even desirable—with a simulation.) That is, the writers felt that a program should rank *high* in flexibility, should call for *active* student involvement, should make *good* use (if any) of color, and should employ competition in a *positive* way only. This left-is-negative and right-is-positive arrangement of the scales allows one to get the general sense of an evaluation by scanning the completed checklist. When items are added or altered on the checklist, it would be a good idea to keep this design intact. Thus, should you feel that program control should be the purview of the teacher or that judgments should be within the program itself, you might reword the second part of item 8 to read as follows:

<i>low</i>					<i>high</i>	
*	*	*	*	*	*	freedom from student control

As rewritten, the word *low* would continue to be associated with an undesirable characteristic.

SECTION 4

Communicating with Posterity: Guidelines for Software Documentation

The Golden Rule

Only some computer-generated humanoid incorporating the joint charms of Scrooge and the Marquis de Sade would lock away program evaluations in some academic safe-deposit box. The long-term value of evaluation guidelines lies in establishing a bank of evaluations that preserves the work for easy retrieval at some future date and that earns interest through the contributions of other professionals. If, of course, you prefer the work of doing each evaluation yourself (and redoing it each time forgetfulness intervenes), this section won't appeal to you.

For the sane and humane among you, the task is clear. We need some simple way of preserving our judgments for both ourselves and our colleagues. The Sample Software Documentation Sheet on page 19 is meant to meet that need. Printed on a single sheet of paper, the sheet summarizes the information gathered from the vendor's documentation (see section 2 of these *Guidelines*) *as modified and expanded by your professional judgment based on your evaluation along the lines of section 3.*

Fact versus Judgment

In the battle of fact and judgment, put your money on judgment. It is the task of professionals to make judgments and their fate to live with them. Facts are of value as means, not ends, of the decision-making process. However, it is our responsibility not to confuse fact with judgment. With the exception of the items regarding Program Source and System Requirements, the completion of the software documentation sheet is almost wholly a judgmental task. The items on the front of the sheet call for the evaluator's informed judgment of how the program might be used, with whom it might be used, and where it should be placed within an instructional sequence. The items on the back ask for descriptions, in terms of the evaluator's views, of the purpose and features of the program as well as for suggestions regarding its use. If the evaluator provides the additional documentation of a sample run, the selection of demonstrative segments will effect a coloring of the evaluation.

Foolish Consistency: Hobgoblin of Little Minds

It is an axiom of the instructional computing devotee that any educational process requiring repetition, speed, or prodigious feats of memory is a candidate for computerization. The use of these guidelines is such a process. Urgings of consistency demand that we consider its computerization. The letters, checklists, documentation sheets, definitions, and procedural steps all could be housed in a computer with the aid of a word-processing program. In that electronic form, they are quickly accessible to any person for use, modification, or recall of the fruits born of their use. Past evaluations can be updated or deleted as software is used, improved, or retired. Data management programs permit the efficient search for programs of potential interest and the recall of the detailed evaluation of a program so identified. Therefore, we conclude this section with the hope that you will transfer the substance of this document to your favorite computer and consign the document itself to the flames.



SAMPLE SOFTWARE DOCUMENTATION SHEET

PROGRAM NAME: _____

SUBJECT AREA: _____

PROGRAM CLASS

- ☐ isolated program
☐ part of a ☐ program cluster

INTENDED AUDIENCE(S)

- ☐ teachers
☐ students in grade(s) _____
☐ other _____

PROGRAM USES: (1=primary; 2=secondary)

- | | |
|---|--|
| ☐ demonstration | ☐ programming utility |
| ☐ drill or practice | ☐ simulation |
| ☐ informational | ☐ testing operations |
| ☐ instructional gaming | ☐ tutorial |
| ☐ instructional management | ☐ whistles and bells |
| ☐ instructional support | ☐ word processing |
| ☐ problem solving | ☐ other (_____) |

CRITICAL PREREQUISITE SKILLS

NAME(S) OF PREREQUISITE PROGRAM(S)

NAME(S) OF FOLLOW-UP PROGRAM(S)

DATE OF THIS DOCUMENTATION

PROGRAM SOURCE

name: _____
address: _____

phone: _____
program cost: _____

SYSTEM REQUIREMENTS

computer: _____
language: _____
memory needed: _____ bytes
input mode: cassette
 _____-inch diskette
 cartridge
 other _____

output mode: color monitor
 B/W monitor
 printer
 plotter
 other _____

CLASSROOM VALIDATION (describe)

TIME FOR AVERAGE EXECUTION

_____ minutes

OVER 

GENERAL DESCRIPTION OF THE PROGRAM (i.e., purpose of the program)

SPECIAL PROGRAM CHARACTERISTICS (e.g., use of graphics or sound, designed for use by groups of two or more persons at a time, level of difficulty can be modified by teacher (only) or student, exit from program is controlled by the teacher (only) or student, etc.)

DIRECTIONS FOR USE OF THE PROGRAM (e.g., directions not presented in the program itself but needed for program execution, directions regarding the use of the program within the total instructional process, notes regarding program options, etc.)

ANCILLARY MATERIALS REQUIRED (e.g., books, worksheets, charts, data lists, dice, geometric shapes, lab equipment, etc.)

ADDITIONAL DOCUMENTATION AVAILABLE (check box for each type)

- ☐ sample run (location _____)
- ☐ program listing (location _____)

SECTION 5

Asides to the Instructional Programmer

The comments and suggestions of this section are directed primarily to the evaluator of software who is also a programmer. For the purpose of this discussion, an *instructional programmer* is one who is capable of producing instructional software either for sale or for exchange with others. A concomitant talent or vice is the ability to alter software written by others. When evaluating software, the instructional programmer should consider both the ease with which a given computer program can be altered and the possible legal issues associated with making any alterations. It is assumed (if untested) that software can be tinkered with *if* it has been purchased and is not to be offered for resale. Nonetheless, an instructional programmer may want to consult with a good lawyer.

Numerous reasons exist as to why one may *want* to change a software program as well as to why it should be changed at all. For example, cosmetic alterations are often desirable and yet minor in terms of the programmer's intent. When more substantial changes are felt necessary or attempted, greater concern is raised.

The Words Have Changed, But I Recognize the Tune

Cosmetic changes are an issue when the instructional programmer likes a particular program on the whole but objects to it in part. The desired alterations may make the program palatable by changing output statements, graphics, or algorithms. For example, the instructional programmer may want to revise the language or comments made in the run of a program. The original form may have been sexist, appeared overly negative in tone, or employed poor vocabulary.

The program may need to be updated. Consider, for example, an economics simulation involving prices, sample products, and interest rates. Such data must be altered periodically if the simulation is to remain current. Finally, one may wish to alter or delete graphic or audio routines that the user may regard as childish or demeaning or that may be recognized as misleading or counterproductive. Thus, the flashing "smiley face" called up by each correct response to a drill program may soon be viewed as silly or phony. Similarly, loud noises generated in response to incorrect answers may begin to grate on the nerves.

As a group, the changes suggested above do not alter the instructional intent of the program's author. In fact, they may serve to support the author's purpose. Things may look a bit different after the program alterations, but the overall development remains intact.

Can You Still Name That Tune?

The instructional programmer may also want to make alterations that are more substantial and of greater impact than those made for cosmetic reasons. It is at this point that the toes of the program's original author begin to ache. Some obvious examples of substantive changes are the correction of known programming flaws, systems errors, poor programming, or inefficient algorithms. A less obvious example is a major alteration of the program's use (or misuse) of color that would move it from a decorative to a functional role.

At still another level, the instructional programmer may contend that the original program has a weak link in its pedagogical sequence and wish to strengthen the program by introducing a totally new sub-routine. Along the same line, the wish may be to alter or rewrite a critical algorithm in an attempt to improve or extend its application.

The alterations commented on in the preceding two paragraphs can result in substantial changes

in the breadth, approach, or effectiveness of the overall program. Where do the editorial privileges of the instructional programmer begin to infringe on the ownership rights of the original author? Who really is the author of the revised program?

The Many Faces of Documentation

When evaluating software with an eye to its possible modification, program documentation becomes critical. There are at least four levels of documentation; the instructional programmer must be careful that the level of the program being evaluated is sufficient to the programmer's purpose. The levels and their potential usefulness are as follows:

1. The documentation may be a brief description of the program's intent, such as that included on an advertisement or on the sample software documentation sheet of section 4. Such statements are of no real value to the instructional programmer desiring to make changes in a program.
2. The documentation may be an extended description of the details of the program's flow, possibly even in some form of flowchart. This category is again of minimal value because of its general nature.
3. The documentation may include the previous category as well as a detailed explanation of key algorithms in the program. This category is of value only if the instructional programmer can determine how each algorithm affects, or relates to, the program as a whole. For example, variables may be present in a routine that later serve as key parameters in either graphics routines or other algorithms.
4. The documentation may be very extensive and complete while also being merged within the listing of the program. It should include the definition of all key variables, descriptions of commands that control the flow of the program, and an elaboration of the steps in each subroutine or algorithm. Furthermore, the name and address of the program's original author and its creation data should also be included to facilitate any communication that may be necessary. Obviously, this category of documentation is of greatest value to the instructional programmer.

As a final concern, the instructional programmer must consider whether the program is of sufficient value to be altered. That is, the purchased form of the program may be in a specially coded form or of such complexity that the instructional programmer may conclude that it would be easier to rewrite the entire program and include the necessary modifications. This raises very real concerns regarding the authorship of the revised program because of related questions regarding whether one can copyright an idea.

SECTION 6

Next Steps

What Follows 6?

We opened our discussion of instructional computing by asking "What Follows 1?" and contended that the answer depended on the computing experiences and interests of the reader. Having guided you to this point by one of five routes, we hope we have stimulated an interest in further exploration of some of the issues overviewed in these *Guidelines*. If so, we would be remiss in not suggesting some other places to visit when this tour is complete. The purpose of this section is to present a brief (and, hence, noncomprehensive) list of books, articles, papers, and bibliographies on four topics of possible interest. The first two sets of references assume no knowledge of computer programming; the latter two are addressed to those readers who either have such knowledge or wish to acquire it.

Computers in the Classroom

Items A through R in our first list of resources deal with instructional uses of computers, both in general and with specific reference to the teaching of mathematics. Items E and P are special-focus issues of the *Mathematics Teacher* (November 1981) and *Arithmetic Teacher* (February 1983); each discusses instructional applications of computers which include *but are not restricted to* mathematics education. Item R is a how-to-do-it guide produced by the experienced classroom teachers who comprised the Montana Task Force on Computer Education.

The remaining selections of the following list provide nontechnical information about instructional computing. These books and monographs are addressed to the person who has been seeking a simple introduction to these topics—one that isn't wrapped up in a course in computer programming.

- A. *Computers and Education*, C. E. Horn and J. Poirot (New York: Sterling Swift Publishing), 1980.
- B. *Computers in the Classroom*, H. Kepner, editor (Washington, DC: National Education Association), 1982.
- C. *Computers in Mathematics: A Sourcebook of Ideas*, D. Ahl, editor (Creative Computing, P.O. Box 789-M, Morristown, NJ 07960).
- D. *Computers in Mathematics Education: 1984 Yearbook*, V. P. Hansen and M. J. Zweng, editors (National Council of Teachers of Mathematics, 1906 Association Drive, Reston, VA 22091), 1984.
- E. *Computing & Mathematics: The Impact on Secondary School Curricula*, J. T. Fey, editor (National Council of Teachers of Mathematics, 1906 Association Drive, Reston, VA 22091), 1984.
- F. *Focus on Computers in Mathematics Education* (National Council of Teachers of Mathematics, 1906 Association Drive, Reston, VA 22091), November 1981.
- G. *Introduction to Computers in Education for Elementary and Middle School Teachers*, D. Moursund (International Council for Computers in Education, 135 Education, University of Oregon, Eugene, OR 97403), 1981.
- H. *Learning with Computers*, Alfred Bork (Billerica, ME: Digital Press), 1981.
- I. *Microcomputers and the 3 R's: A Guide for Teachers*, C. Doerr (Rochelle Park, NJ: Hayden Book Co.), 1980.
- J. *Microcomputers in Education: An Introduction*, A. Naiman (Technical Education Research Centers, 8 Eliot Street, Cambridge, MA 02138), 1982.
- K. *Microcomputers for Teachers—with Application to Science and Mathematics*, D. Zalewski, editor, Topics for Teachers Series (School Science and Mathematics Association, 126 Life Science Building, Bowling Green State University, Bowling Green, OH 43403), 1982.
- L. *Mindstorms: Children, Computers, and Powerful Ideas*, S. Papert (New York: Basic Books), 1980.

- M. *Peanut Butter and Jelly Guide to Computers*, J. Willis (Forest Grove, OR: Dilithium Press), 1979.
- N. *RUN: Computer Education*, D. O. Harper and J. H. Stewart (Monterey, CA: Brooks/Cole Publishing Co.), 1983.
- O. *Teachers Guide to Computers in the Elementary School*, D. Moursund (International Council for Computers in Education, 135 Education, University of Oregon, Eugene, OR 97403), 1982.
- P. *Teaching with Microcomputers* (National Council of Teachers of Mathematics, 1906 Association Drive, Reston, VA 22091), February 1983.
- Q. *The Computer in the School: Tutor, Tool, Tutee*, R. Taylor, editor (New York: Basic Books), 1980.
- R. *The Elements of Computer Education: A Complete Program* (Office of Public Instruction, State Capitol, Helena, MT 59620).
- S. *The Many Faces of Instructional Computing: An Actions Guide for Educators*, J. R. Dennis and B. Kansky (Glenview, IL: Scott, Foresman and Co.), 1984.

Computers Everywhere Else

The second set of resources examines the social and moral issues associated with the current and projected impact of computer technology upon society. The listing is modest and, considering the rapidly changing nature of that technology, dated. New books (and some articles) on the topic can be found by searching the bibliographies of Item B. For even more up-to-date thinking about the societal side of computing, consult the indexes of journal articles (G) under such headings as "Computer Literacy" or "Computers and Society."

- A. *Are You Computer Literate?* K. Billings and D. Moursund (Forest Grove, OR: Dilithium Press), 1979.
- B. Bibliographies
 - 1. *Annual Bibliography of Computer-oriented Books* (Computing Newsletter, P.O. Box 7345, Colorado Springs, CO 80933).
 - 2. "Computer Literacy Bibliography," S. Friel and N. Roberts (*Creative Computing*, September 1980, pp. 92-97).
 - 3. *The Impact of Computers on Society and Ethics: A Bibliography*, G. Abshire (Creative Computing, P.O. Box 789-M, Morristown, NJ 07960), 1980.
 - 4. *The Prospect of Computers on Society and Ethics: A Bibliography*, G. Abshire (Morristown, NJ: Creative Computing Press), 1980.
- C. *Computers and Their Societal Impact*, M. Holvien (New York: John Wiley & Sons), 1979.
- D. *Computers and Social Controversy*, T. Logsdon (Potomac, MD: Computer Science Press), 1980.
- E. *Computers and Society*, S. Rothman and C. Mosmann (Chicago: Science Research Associates), 1976.
- F. *Computer Consciousness: Surviving the Automated 80's*, H. D. Covey and N. H. McAlister (Reading, MA: Addison-Wesley Publishing Co.), 1980.
- G. Journal Indexes
 - 1. *COMPendium* (Epicurious, P.O. Box 129, Lindendale, NY 10540).
 - 2. *Microcomputer Index* (Microcomputer Information Services, 2464 El Camino Real, Suite 247, Santa Clara, CA 95051).
 - 3. *Micro . . . Publications in Review* (Vogeler Publishing, P.O. Box 489, Arlington Heights, IL 60006).
 - 4. *The Index* (W. H. Wallace, Missouri Indexing, P.O. Box 301, Saint Ann, MO 63074).
- H. *Running Wild: The Next Industrial Revolution*, A. Osborne (Berkeley, CA: Osborne/McGraw-Hill), 1979.
- I. *Tales of the Marvelous Machine*, R. Taylor and B. Green (Morristown, NJ: Creative Computing Press), 1980.
- J. *The Compleat Computer*, D. Van Tassel (Chicago: Science Research Associates), 1976.
- K. *The Computers in Our Society and The Computers in Our Society Workbook*, T. Logsdon and F. Logsdon (Fullerton, CA: Anaheim Publishing Co.), 1979.
- L. *The Micro Millenium*, C. Evans (New York: Washington Square Press), 1981.

Back to BASICS

The computing language most commonly taught in precollege classrooms at the time of this writing is BASIC, although there is developing interest in Logo for the elementary and middle schools, and in Pascal in the secondary schools, particularly for students who are in college-preparatory curricula. Again we have a situation in which whatever we say at this stage, it will be out of tune almost by the time the ink is dry. Each of the languages has advantages and disadvantages depending on the purposes for which it is being used. The following discussion, however, is focused only on BASIC for two reasons: (1) it is currently the most commonly encountered language across the precollege levels and (2) stock models of current versions of school computers come equipped to communicate in BASIC (with communication in Logo or Pascal available as an add-on).

BASIC exists in many versions, each of which is evolving through constant revision. Despite these variations, there are sufficient fundamental similarities among these versions to permit one to speak of "the" BASIC language with but a slight wince.

Numerous books introduce the fundamental concepts of programming in BASIC. Some are produced by the manufacturers of a particular computing system and teach the version of BASIC specific to that system. The books listed below are simply *samples* (i.e., ones familiar to the authors). Many other excellent texts—some developed for such special audiences as elementary school students—are reviewed monthly in the journals of professional associations and in computing journals.

Ten of the fourteen items in our list are books that teach the "fundamental similarities" alluded to earlier, occasionally commenting on variations regarding a particular BASIC concept. Items A and C present a "soft," low-level introduction to programming in BASIC. Item B elaborates on the introductory work of item A; the result is an extensive course in beginning programming for persons from a wide range of subject matter fields. The texts listed in items D, E, J, and L promote the structured style of programming that is so dear to the heart of computerists (and which will capture your academic affection as soon as you've tried it). Four of the books (D, E, F, and G) give special attention to programming mathematical topics.

The remaining four items of the list deserve special mention. Items H and N could be called "companion" texts for a course in BASIC. In fact, although item H is usually associated with the study of Pascal, its use might precede the study of any particular language, since it develops several fundamental programming principles without so much as making use of the concept of a variable. Item N would complement most textbooks in BASIC instruction by helping the learner to write programs in a style that is easy to read (and, hence, to debug). Once a *version* of BASIC has been mastered, item M will prove an invaluable reference text, since it is a handy compendium of the commands and statements found in several versions of the BASIC language. Finally, the articles listed in item I allow the learner to stand back from the details of a particular computing language and to consider computing languages as a generic concept.

- A. *A Bit of BASIC*, T. Dwyer and M. Critchfield (Reading, MA: Addison-Wesley Publishing Co.), 1980.
- B. *BASIC and the Personal Computer*, T. Dwyer and M. Critchfield (Reading, MA: Addison-Wesley Publishing Co.), 1978.
- C. *BASIC: A Hands-On Approach*, H. Peckham (New York: McGraw-Hill Book Co.), 1981.
- D. *BASIC: A Structured Approach*, C. Siragusa (Boston: Prindle, Weber & Schmidt), 1980.
- E. *BASIC Discoveries: A Problem-solving Approach to Beginning Computer Programming*, L. Malone and J. Johnson (Palo Alto, CA: Creative Publications), 1981.
- F. *Computer Programming in BASIC*, T. P. Pavlovich and T. E. Taham (San Francisco: Holden-Day), 1971.
- G. *Computer Programming in the Basic Language*, 2d ed., N. Golden (New York: Harcourt Brace Jovanovich), 1980.
- H. *Karel the Robot*, R. Pattis (New York: John Wiley & Sons), 1981.
- I. Languages in General
 - 1. "A Beginner's Guide to Programming Languages," S. Hughes (*On Computing*, Spring 1980, 52-59).
 - 2. "Programming Languages," J. Feldman (*Scientific American*, December 1979, 94-116).
 - 3. "What Is a Programming Language?" A. Wold (*Classroom Computer News*, April 1983, 46-49).

- J. *Problem Solving and Structured Programming in BASIC*, E. Koffman and F. Friedman (Reading, MA: Addison-Wesley Publishing Co.), 1979.
- K. *Programming for Computer Literacy*, D. Moursund (New York: McGraw-Hill Book Co.), 1978.
- L. *Programming Primer*, R. P. Taylor (Reading, MA: Addison-Wesley Publishing Co.), 1982.
- M. *The BASIC Handbook: An Encyclopedia of the BASIC Computer Language*, D. A. Lien (San Diego: COMPUSOFT Publishing), 1978.
- N. *The Little Book of BASIC Style*, J. Nevison (Reading, MA: Addison-Wesley Publishing Co.), 1978.

The All-Purpose Tool

The final collection of references reveals the value of the computer as a versatile servant of the mathematical problem solver. Four (B, D, E, and F) apply the computer to problems in precalculus mathematics, one (A) presents applications in calculus, and one (C) fully integrates the computer into the study of elementary statistics.

- A. *Computer Applications for Calculus*, W. S. Dorn, G. Bitter, and D. L. Hector (Boston: Prindle, Weber & Schmidt), 1972.
- B. *Computer Resource Book: Algebra*, T. Dwyer and M. Critchfield (Boston: Houghton Mifflin Co.), 1975).
- C. *Elementary Computer-assisted Statistics*, F. Scalzo and R. Hughes (New York: Petrocelli/Charter), 1976.
- D. *Mathematical Problem Solving with the Microcomputer*, L. Snover and M. Spikell (Englewood Cliffs, NJ: Prentice-Hall), 1982.
- E. *Problems for Computer Solution*, S. Rogowski (Morristown, NJ: Creative Computing Press), 1975.
- F. *Using Computers in Mathematics*, G. Elgarten (Menlo Park, CA: Addison-Wesley Publishing Co.), 1983.

SECTION 7

Extra Forms

The forms on the next two sheets are copies of those presented earlier in this booklet. They can be removed and reproduced as needed.

The NCTM gladly grants the reader permission to reprint an unlimited number of these forms for nonprofit purposes. It requests that the NCTM logo be retained if the forms are used without substantive alteration of their content. The user is also invited to modify the documents in any way that might improve their usefulness.



SOFTWARE EVALUATION CHECKLIST

PROGRAM NAME: _____ SOURCE: _____ COST: _____

SUBJECT AREA: _____ REVIEWER'S NAME: _____ DATE: _____

1. INSTRUCTIONAL RANGE

_____ grade level(s)

_____ ability level(s)

2. INSTRUCTIONAL GROUPING FOR PROGRAM USE

_____ individual

_____ small group (size: _____)

_____ large group (size: _____)

3. EXECUTION TIME

_____ minutes (estimated) for average use

4. PROGRAM USE(S)

_____ demonstration

_____ drill or practice

_____ instructional gaming

_____ instructional management

_____ instructional support

_____ problem solving

_____ programming utility

_____ simulation

_____ testing operations

_____ tutorial

_____ whistles and bells

_____ word processing

_____ other (_____)

5. USER ORIENTATION: INSTRUCTOR'S POINT OF VIEW

low

high

* * * * * flexibility

* * * * * freedom from need to intervene or assist

6. USER ORIENTATION: STUDENT'S POINT OF VIEW

low

high

* * * * * quality of directions (clarity)

* * * * * quality of output (content and tone)

* * * * * quality of screen formatting

* * * * * freedom from need for external information

* * * * * freedom from disruption by system errors

* * * * * simplicity of user input

7. CONTENT

low

high

* * * * * instructional focus

* * * * * instructional significance

* * * * * soundness or validity

* * * * * compatibility with other materials used

8. MOTIVATION AND INSTRUCTIONAL STYLE

passive

active

* * * * * type of student involvement

low

high

* * * * * degree of student control

none poor

good

* * * * * use of game format

* * * * * use of still graphics

* * * * * use of animation

* * * * * use of color

* * * * * use of voice input and output

* * * * * use of nonvoice audio

* * * * * use of light pen

* * * * * use of ancillary materials

* * * * * use of _____

9. SOCIAL CHARACTERISTICS

*present and
negative*

*not
present*

*present and
positive*

_____ competition

_____ cooperation

_____ humanizing of computer

_____ moral issues or value judgments

_____ summary of student performance

Key to Terms: Software Evaluation Checklist

1. The grade levels and ability levels for a particular program are primarily determined by the concepts involved. Other important factors are reading level, prerequisite skills, degree of student control, and intended instructional use. It is possible for a program to be flexible enough to be used across a wide range of grade levels and ability levels.

2. Some programs are designed for use by individuals. Others have been or can be modified for participation by two or three persons at a time. Simulations or demonstrations often pose opportunities for large group interaction. A given program may be used in more than one grouping, depending on the instructor.

3. The time required for the use of a program will vary considerably. Include loading time for cassettes. A time range is the appropriate response here.

4. Instructional programs can be categorized according to their uses. Some programs may have more than one use, thus falling into more than one of the following categories:

Demonstration: Used by a teacher for leading group discussion.

Drill or practice: Assumes that the concept or skill has been taught; merely reinforces.

Informational: Generates data but does not direct the use of those data.

Instructional gaming: Focuses on decision making within a game format; involves random events and the pursuit of a winning strategy.

Instructional management: Helps manage the learning environment by assuming major responsibility with respect to student assessment, record keeping, and generating learning prescriptions.

Instructional support: Generates instructional materials, helps locate instructional resources, or aids in selecting instructional products.

Problem solving: An algorithm that serves the problem solver during the exploration of a complex problem.

Programming utility: Programs that help write other instructional programs.

Simulation: An electronic slice of life.

Testing operations: Provides services such as item banking, test generation, test scoring, and item analysis.

Tutorial: A dialogue between student and computer on a topic that is not yet understood by the student.

Whistles and bells: Demonstrates features of the computer system—features that may be of value in instructional applications.

Word processing: Permits the efficient improvement of written materials.

5. These are factors relevant to the actual use of the program from the point of view of an instructor.

Flexibility: A program may allow the user or the instructor to adjust the program to different ability levels, degrees of difficulty, or concepts.

Intervention or assistance: A rating of "low" means considerable teacher intervention or assistance is required.

6. These are factors relevant to the actual use of the program from the point of view of a student.

Directions: The directions should be complete, readable, under the user's control (e.g., should not scroll off the screen until understood), and use appropriate examples.

Output: Program responses should be readable, understandable, and complete. If in response to student input, the output should be of an acceptable tone and consistent with the input request.

Screen formatting: The formats during a program run should not be distracting or cluttered. Labels and symbols should be meaningful within the given context.

External information: A program may require the user to have access to information other than that provided within it. This may include prerequisite content knowledge or knowledge of conventions used by the program designer as well as maps, books, models, and so on.

System errors: System errors result in the involuntary termination of the program.

Input: The user should know when, where, and in what form input is expected. Student input rather than automatic timing should prompt the presentation of new material on the screen, and specific screen areas should be reserved for such input. Characters with special meanings should be avoided; however, when they are needed, such symbols should be accompanied by explicit instruction and by a readily available "help" display regarding their use. The complexity of typed input should be minimized but should be sufficient to serve the learning objectives of the program.

7. These are matters relevant to the subject-matter content of the program.

Focus: The program topic should be clearly defined and of a scope that permits thorough treatment.

Significance: The instructional objectives of the program must be viewed as important by the instructor. Also, the program should represent a valid use of the computer's capabilities while improving the instructional process.

Soundness or validity: The concepts and terms employed should be correct, clear, and precise. Other important factors are the rate of presentation, degree of difficulty, and internal consistency.

Compatibility: The content, terminology, teaching style, and educational philosophy of the program should not conflict with those generally encountered by the student.

8. These items are self-explanatory.

9. Competition, cooperation, and values are concerns that may be a function of the way a program expresses them. (War gaming and the "hangman" format are sample issues.) Also, the "humanizing" of the computer may serve for motivation or to reduce anxiety, but it also may become tedious, misleading, and counterproductive.

The summary of student performance can be dichotomous (win or lose), statistical (time expended or percent of items correct), or subjective (as in the evaluation of a simulation). It may be for student, teacher, or both.



SAMPLE SOFTWARE DOCUMENTATION SHEET

PROGRAM NAME: _____

SUBJECT AREA: _____

PROGRAM CLASS

- ☐ isolated program
☐ part of a ☐ program cluster

INTENDED AUDIENCE(S)

- ☐ teachers
☐ students in grade(s) _____
☐ other _____

PROGRAM USES: (1=primary; 2=secondary)

- | | |
|---|--|
| ☐ demonstration | ☐ programming utility |
| ☐ drill or practice | ☐ simulation |
| ☐ informational | ☐ testing operations |
| ☐ instructional gaming | ☐ tutorial |
| ☐ instructional management | ☐ whistles and bells |
| ☐ instructional support | ☐ word processing |
| ☐ problem solving | ☐ other (_____) |

CRITICAL PREREQUISITE SKILLS

NAME(S) OF PREREQUISITE PROGRAM(S)

NAME(S) OF FOLLOW-UP PROGRAM(S)

DATE OF THIS DOCUMENTATION

PROGRAM SOURCE

name: _____
address: _____

phone: _____
program cost: _____

SYSTEM REQUIREMENTS

computer: _____
language: _____
memory needed: _____ bytes
input mode: ☐ cassette
 ☐ -inch diskette
 cartridge
 other _____
output mode: ☐ color monitor
 ☐ B/W monitor
 ☐ printer
 ☐ plotter
 other _____

CLASSROOM VALIDATION (describe)

TIME FOR AVERAGE EXECUTION

_____ minutes

GENERAL DESCRIPTION OF THE PROGRAM (i.e., purpose of the program)

SPECIAL PROGRAM CHARACTERISTICS (e.g., use of graphics or sound, designed for use by groups of two or more persons at a time, level of difficulty can be modified by teacher (only) or student, exit from program is controlled by the teacher (only) or student, etc.)

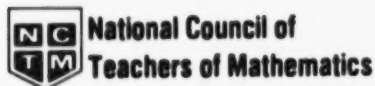
DIRECTIONS FOR USE OF THE PROGRAM (e.g., directions not presented in the program itself but needed for program execution, directions regarding the use of the program within the total instructional process, notes regarding program options, etc.)

ANCILLARY MATERIALS REQUIRED (e.g., books, worksheets, charts, data lists, dice, geometric shapes, lab equipment, etc.)

ADDITIONAL DOCUMENTATION AVAILABLE (check box for each type)

- ☐ sample run (location _____)
- ☐ program listing (location _____)

A position statement of the



COMPUTERS IN THE CLASSROOM

Mathematics programs must take full advantage of the power of computers at all grade levels. Students and teachers should obtain a working knowledge of how one interacts with computers and uses their capacities.

Computer literacy is an essential outcome of contemporary education. Each student should acquire an understanding of the versatility and limitations of the computer through firsthand experiences with applications in a variety of subject matter fields. Mathematics programs designed to take full advantage of the multidisciplinary potential of computers should include the following:

- Problem solving
- Simulations that give the opportunity to practice decision making
- Lessons that introduce and develop concepts
- Word processing to develop communication skills
- Data retrieval tasks such as career information services
- Simulations that replace dangerous, expensive, or technically difficult laboratory work
- Programming
- Drill-and-practice programs
- Student-operated computer services within the community
- Teacher-support services
- Functions that improve the evaluation process

Curriculum materials should be developed that capitalize on unique characteristics of the computer. Such materials should provide instructional experiences heretofore impossible as well as imaginative approaches to existing components of the curriculum.

The educational impact of computers is a community concern. Comprehensive planning is a joint responsibility of teachers, administrators, and parents. Each has a vital role in the definition of computer uses, the development and evaluation of software, and the selection of equipment.

(October 1980)